NASA
Marshall Space Flight Center

Space Flight Software Development

10/20/2004

Luis Trevino
Tim Crumbley

NASA/MSFC Vision

To improve life here,
To extend life to there,
To find life beyond.

NASA's Mission

To understand and protect our home planet,
To explore the Universe and search for life,
To inspire the next generation of explorers…
as only NASA can.

ISS009E24868

MSFC Flight Software

The Flight Software Branch develops software for:

Embedded Space Flight Systems

Real-Time Control Systems

Rocket Engine Controllers

Propulsion Control

Video Guidance Sensors

Space Transportation Vehicles

Spacecraft Control

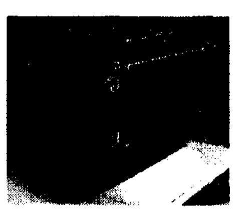
Space Station Systems

Health Monitoring Systems

Microgravity Facilities

Science Experiments

Space Station Payloads



Space Shuttle Main Engine Controller

Mission Critical Space Flight Software Development

Advanced Video Guidance System (AVGS) for DART
Orbital Express AVGS
Microgravity Experiments
Space Station Payload Facility
Space Station Environmental Control System
Propulsion Engine Controllers

Software Technical Insight

Requirements for Human Rated Software
Space Shuttle Main Engine (SSME)
SSME Advanced Health Monitoring
Test Procedures and Facility Software
Engine Controller Software
Autonomous Vehicle Software
Spacecraft Software
Space Station Element Software
Space Station Payload Software
Scientific Instrument Software
Autonomous Docking and Rendezvous Software

Advanced Software Development Technologies

Web-based Software Coding Standard Checker
Software Complexity Analysis Tool
Software Tool Research
Software Metrics Tools
Health Monitoring Software
Use of Generic Algorithms
Applying Soft Computing to Engine Control Software

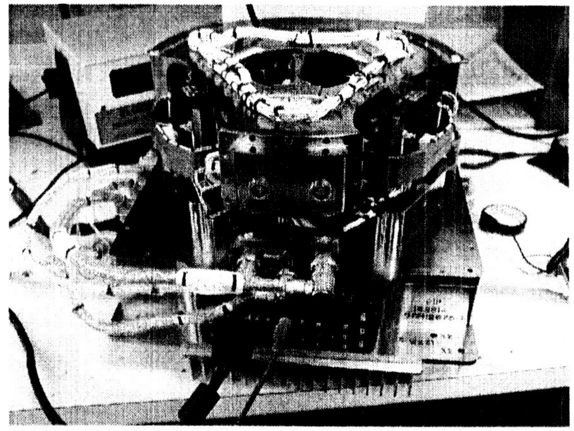
Continuous Improvement in the Software Development Processes and Methods

Software Acquisition Improvement
Agency Software Engineering Requirements
Software Development and Organizational Metrics
Industry Best Practices
Process Asset Library
Industry Process Improvement Model
Defect Metrics
Productivity Data and Metrics
Process Improvement
Risk Management
Software Costing Models
Software Engineering Training

Flight Software Branch

Mission Critical Space Flight Software Development

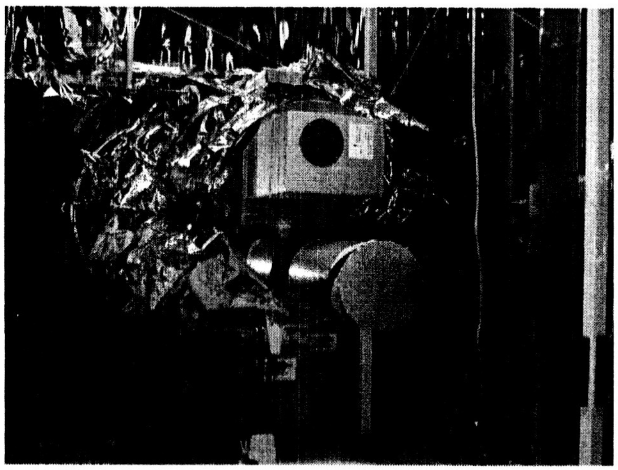
- Software is critical in any Space Launch and Transportation System
- With modern systems, the unique functions are increasingly embodied in the software
- Software engineering has provided missions with capabilities that would not be practical with any other technology



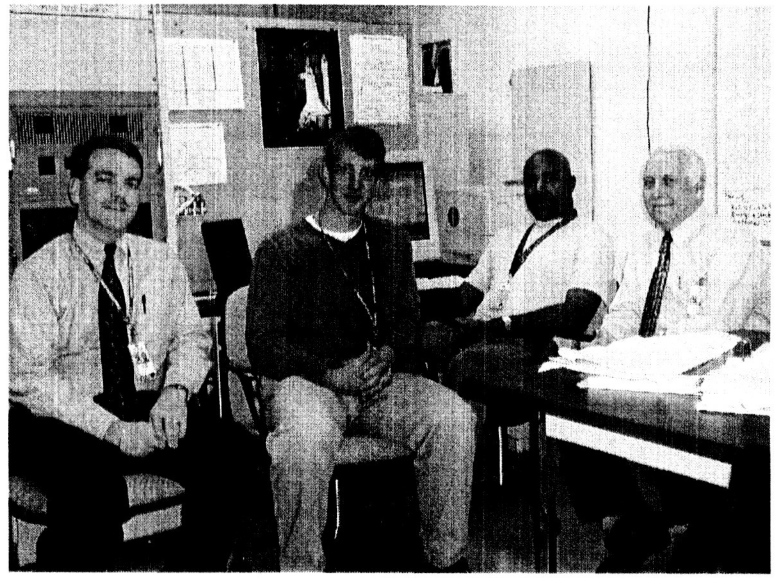
gLIMIT Flight Unit

Space Shuttle Main Engine Software

Over 100 STS missions supported with zero in-flight software anomalies



AVGS on DART



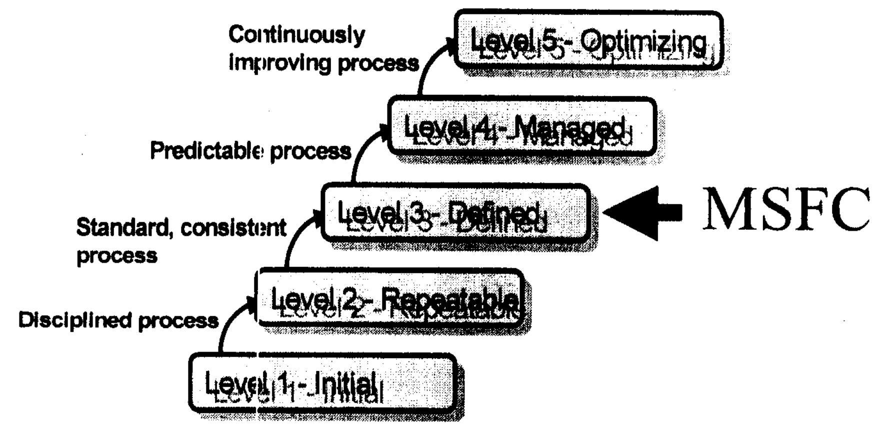
UPA Flight Software Development Team

Flight Software Branch

Continuous Improvement in the software development processes and methods

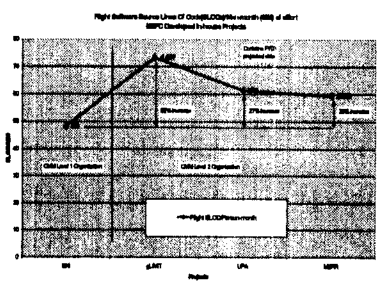
- Currently using the DOD model for benchmarking our software process improvement activities
- Leading the agency in Software Process Improvement

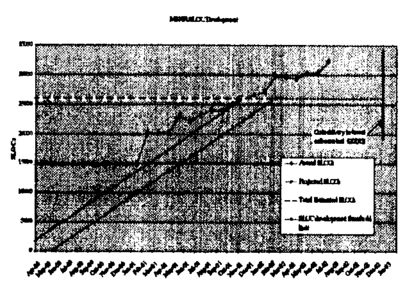
Capability Maturity Model (CMM)



Software metrics

We are using Software development and organizational metrics to manage our software projects





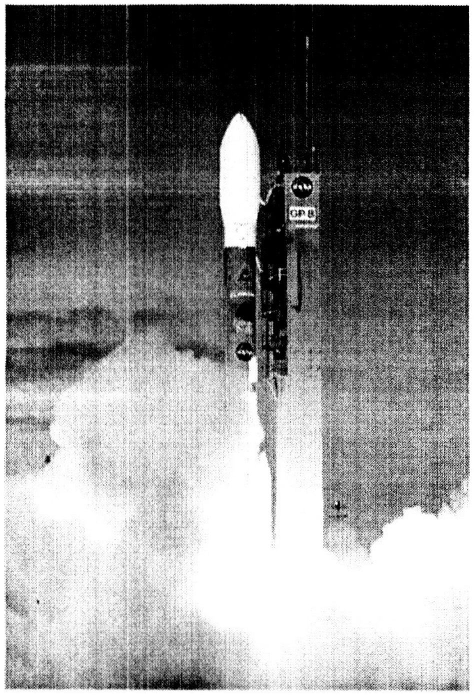
Flight Software Group Goal

To improve our software development processes to enable the Flight Software Group to develop higher quality flight software

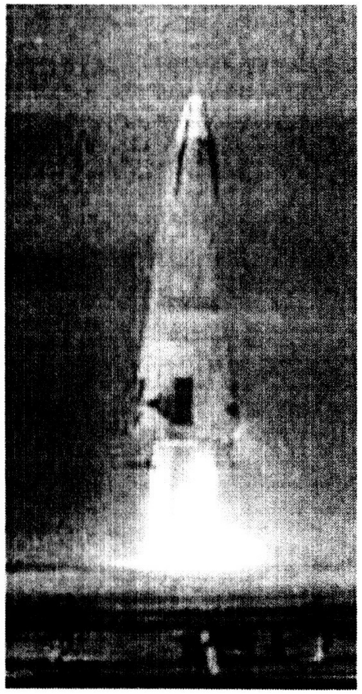
- Producing software with fewer defects
- Producing better schedule estimations
- Providing better resource estimations
- Increase software productivity

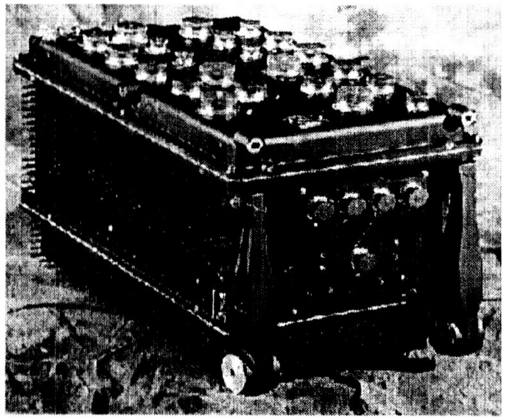
Profile of programming languages used on MSFC Flight Software Projects

Programming Languages	# of MSFC Projects
C	39
Ada	9
Assembly	5
C++	5
Forth 83	2

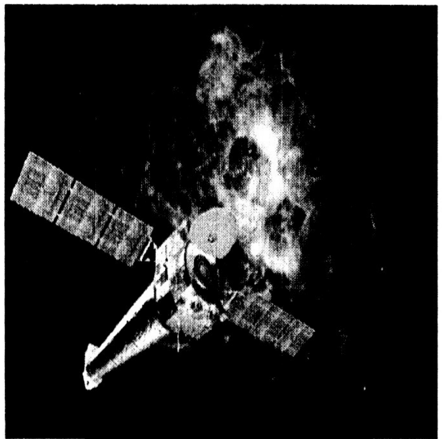


GPB Launch



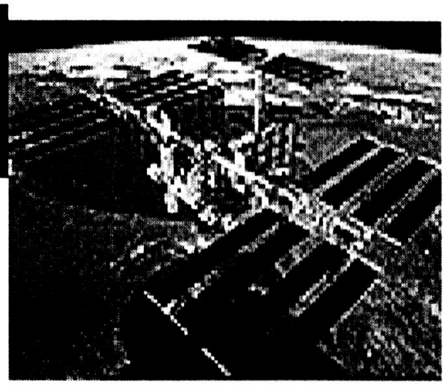


Space Shuttle Main Engine Controller (SSMEC)

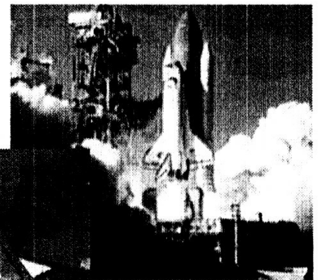


[redacted]

[redacted]

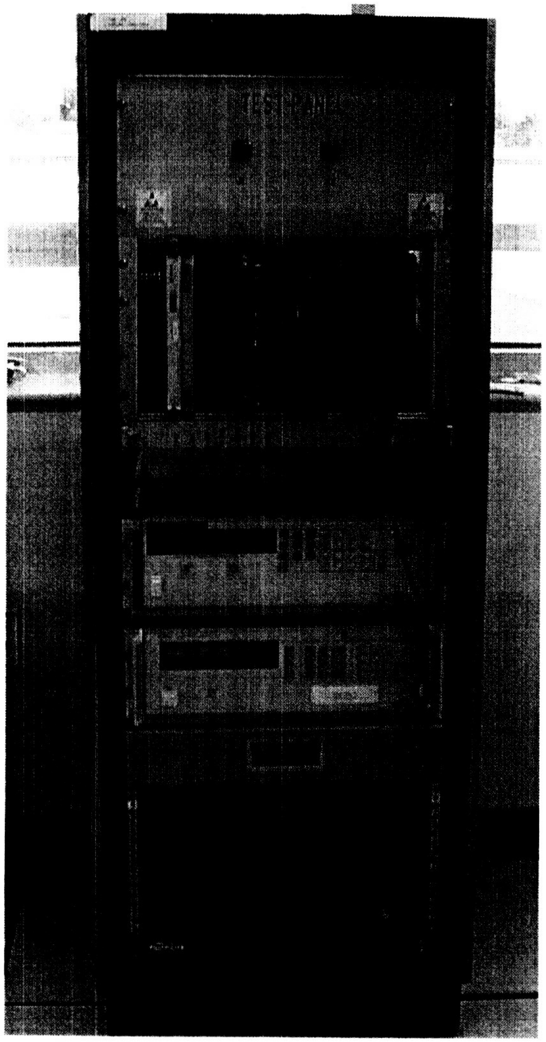


[redacted]

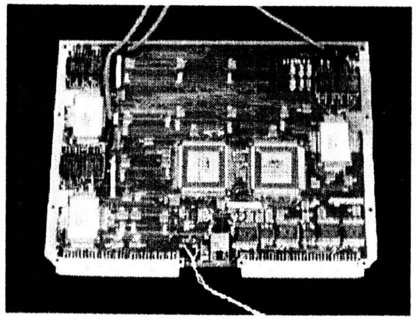


[redacted]

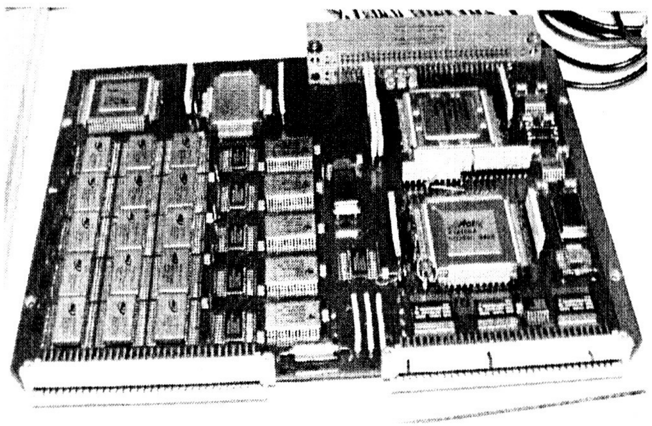
Example Flight & Ground Products



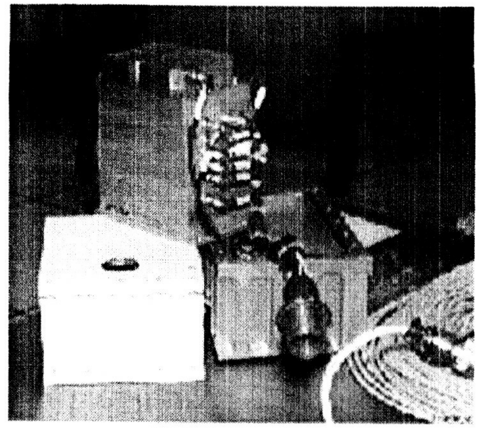
HMC EU Rack



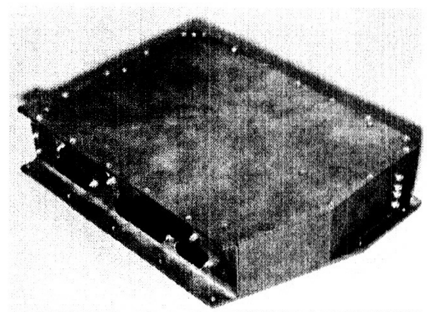
MSRR 1553 Board



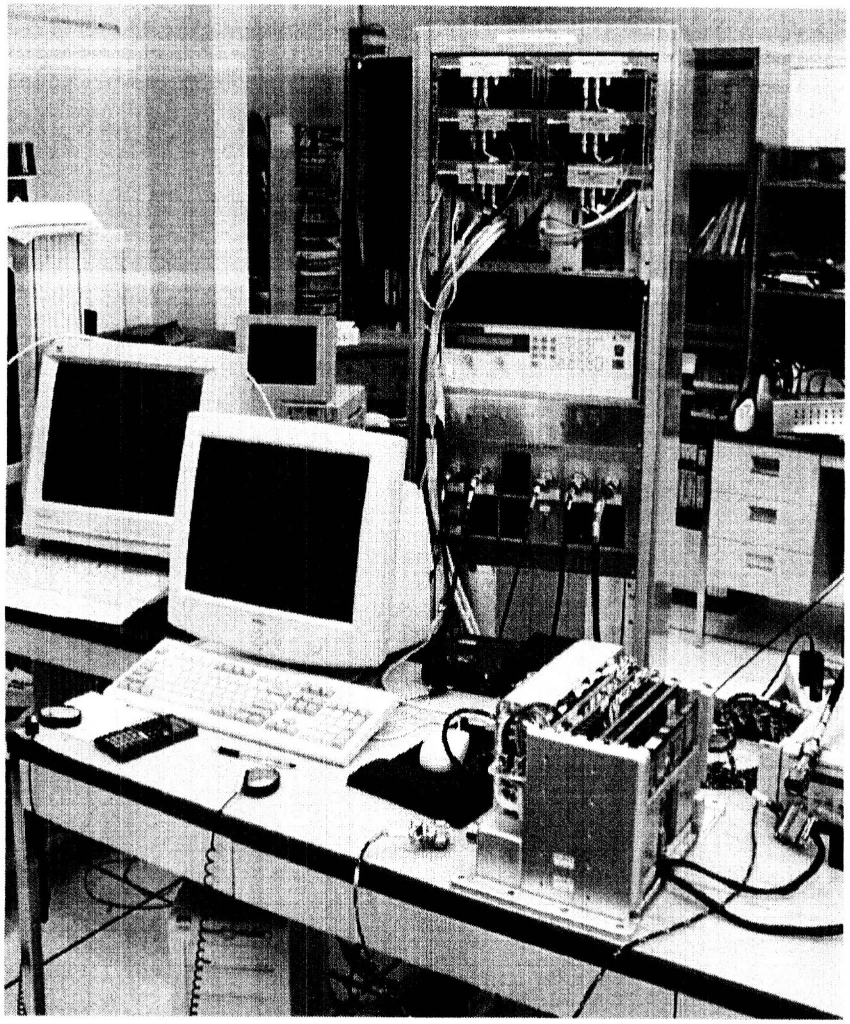
UPA Processor Board



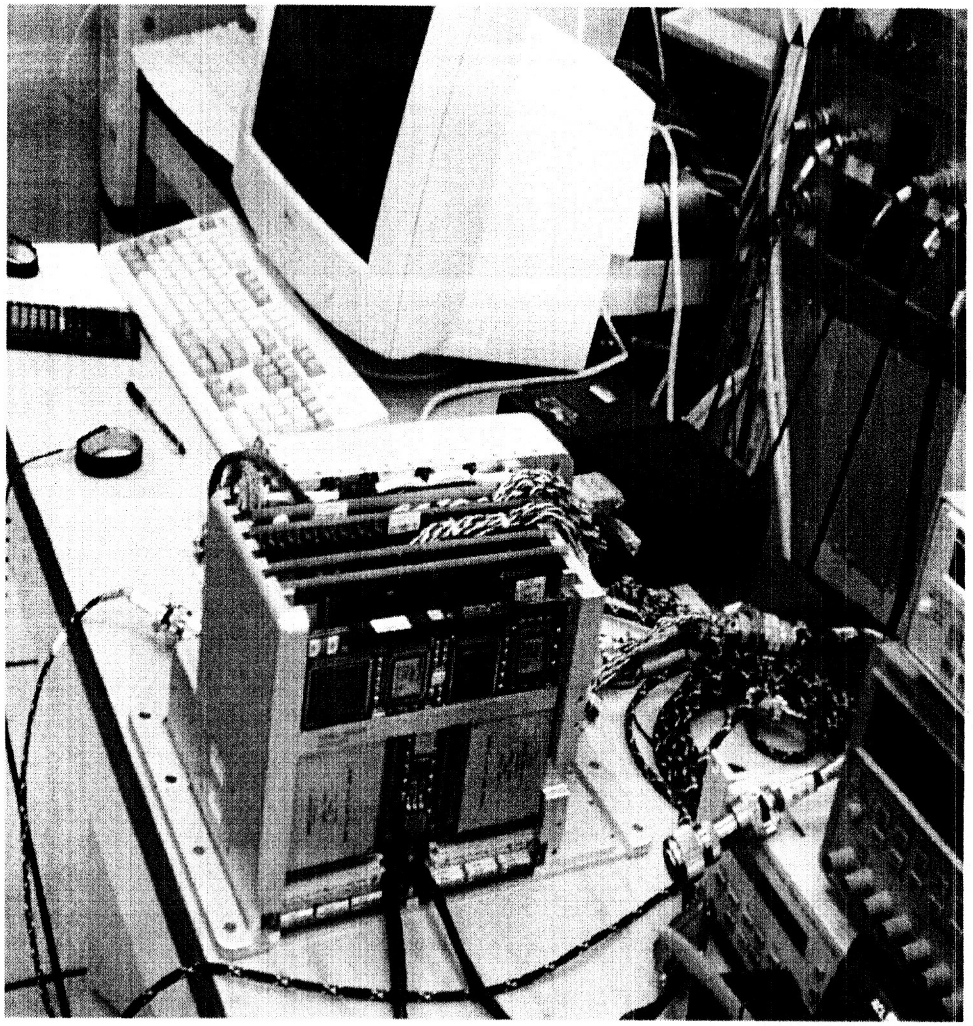
EDAS SRB Hardware



SXI Data
Electronics Box



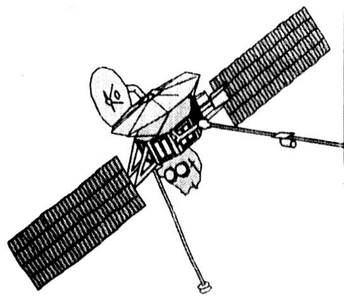
MSRR EU Master Controller & Unit Tester



MSRR EU Master Controller

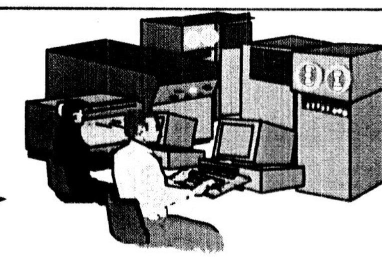
Mission Software: Architecture

End-to-end data systems engineering of mission systems

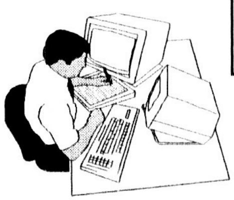


Embedded spacecraft, instrument and hardware component software

Real-time ground mission data systems for spacecraft integration and on-orbit ops (e.g., S/C command & control, launch and tracking services)

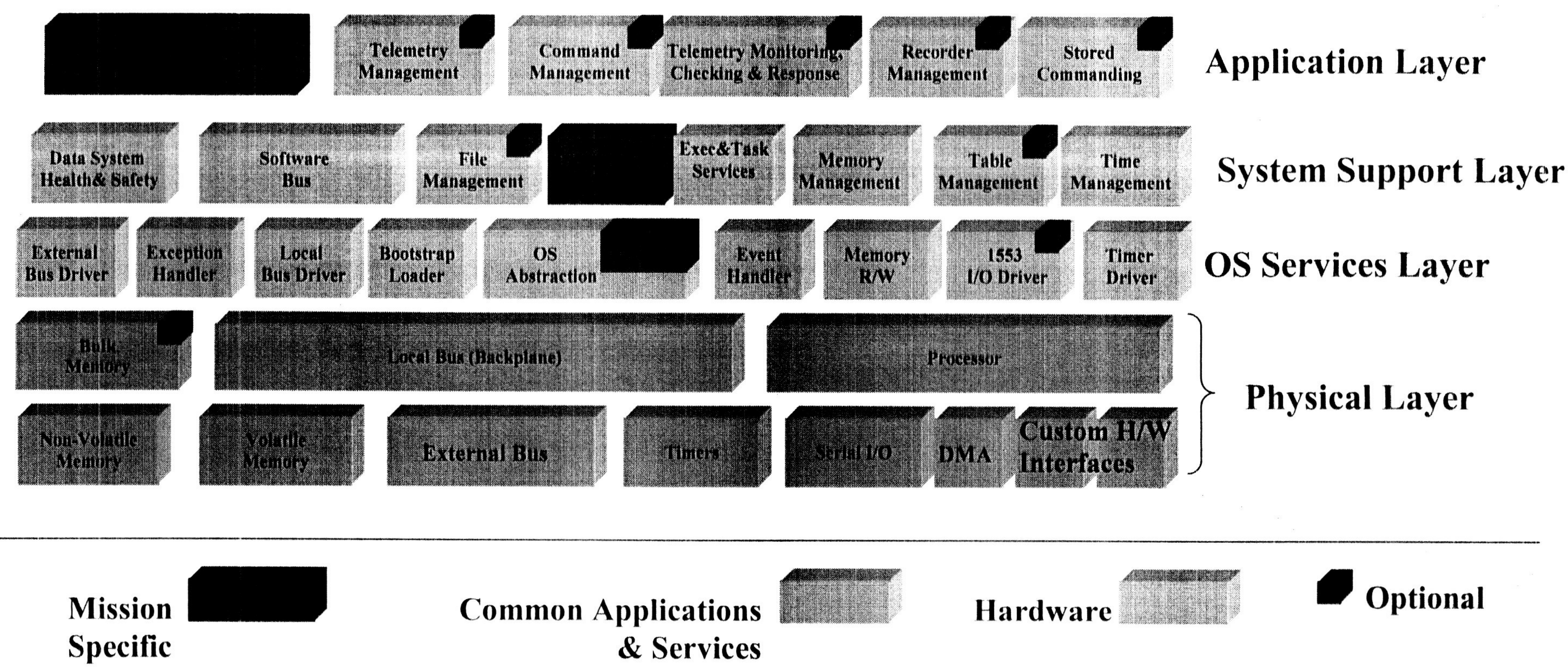


Science data systems including data processing, archival, distribution, analysis & info mgmt.



Off-line mission data systems (e.g., command mgmt., S/C mission and science planning & scheduling, guidance & navigation, network scheduling)

Flight Software Layered Architecture



Lower layers provide portable interface: This enables higher reuse of applications

Different Domains of Software Each Reflect A Different Emphasis

Flight Software

- driven by limited S/C life, asset survival, & mission science program
- continuous critical real-time ops, from attitude control to H&S monitoring
- fixed & constrained environment
- minimize risk with a never fail mindset
- restricted maintenance opportunities

Mission Control Ground Systems

- driven by limited S/C life, asset health, and observatory user demands
- episodic real-time & near-time ops, from command uplink to system state evaluations
- open to needs based augmentation
- risk adverse with a fail soft/over mindset
- full shadow maintenance capability

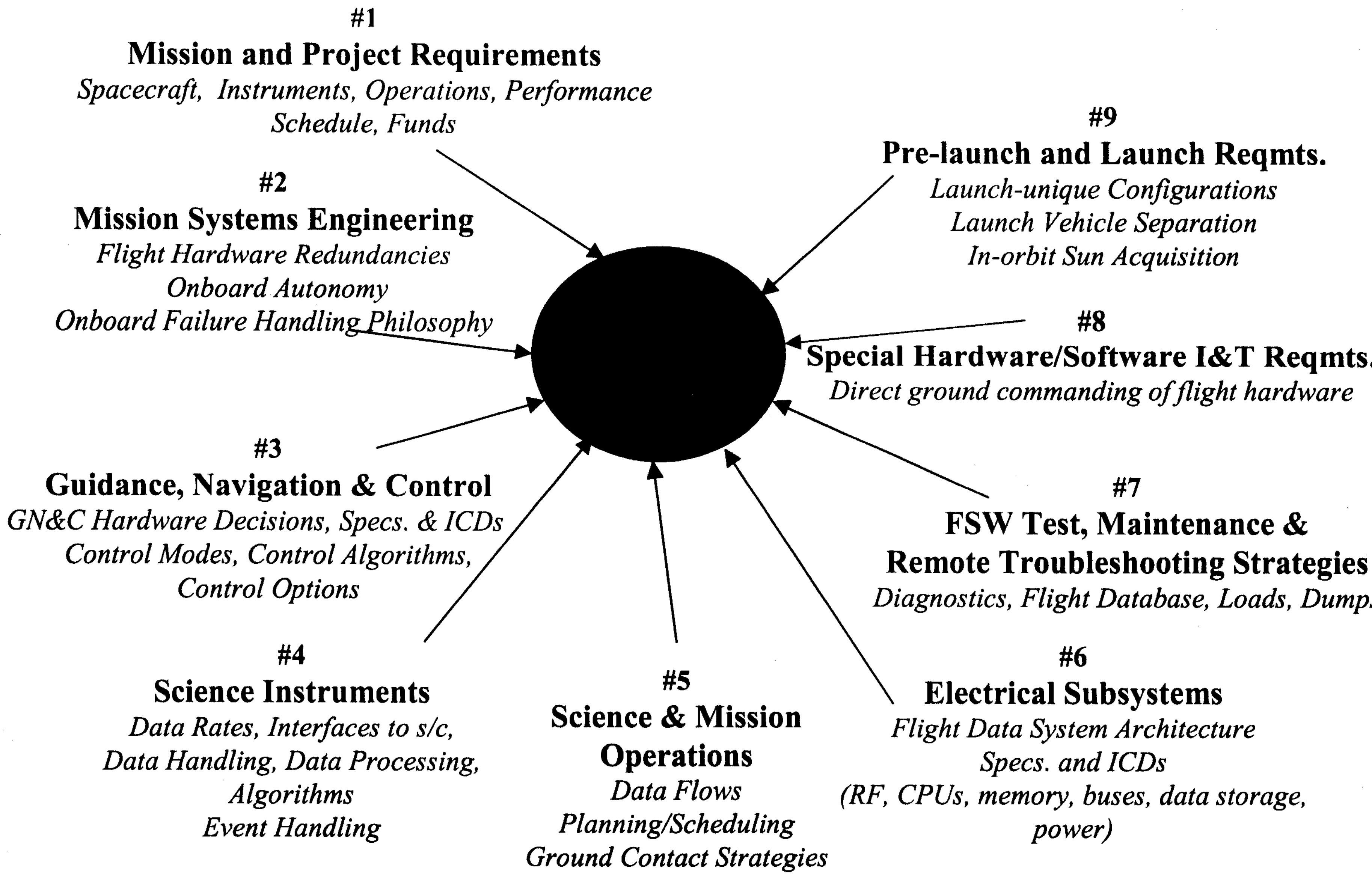
Science Data Management & Data Processing

- data retention & integrity driven
- near-time and later ops, from raw archival to signature calibrations and analysis
- flexible & extendable environment
- data fail soft mindset
- - shadow mode and add-on maintenance

Science Data Dissemination

- science evolution & user driven
- near-time and later ops
- large user communities
- evolving user interfaces & access demands
- timely data delivery mindset
- shadow mode and add-on maintenance

Flight Software Requirements Drivers



Project Management Considerations: Key Documents / Deliverables (more than code)

- Software life-cycle products include
  - Product Plan / Software Management Plan
  - Software Requirements Document (SRD)
  - Interface Requirement Document(s) (IRDs) & Interface Control Documents (ICDs)
  - Software Requirements traceability matrix
  - Functional and Detailed Design Documents
  - Source code
  - Software Test Plan(s)
  - Test and Verification Matrix
  - Software User's Guide
  - Release Letters
  - Delivery, Installation, Operations, and Maintenance Plan(s)
  - Configuration Management Plan
  - Risk Management Plan
  - Software Assurance Plan
  - Software Safety Plan

Software Metrics Approach

- Standard Management principle related to metrics
  - What gets measured gets managed.
  - What gets managed gets done.
  - However, what does not get measured and managed often gets ignored.
- Three basic building blocks of any successful measurement program
  - Measure is a consistent but flexible process that must be tailored to the unique information needs and characteristics of a particular project or organization.
  - Decision makers must understand what is being measured.
  - Measurements must be used to be effective. The measurement program must play a role in helping decision makers understand project and organization issues.

Metrics Management Process

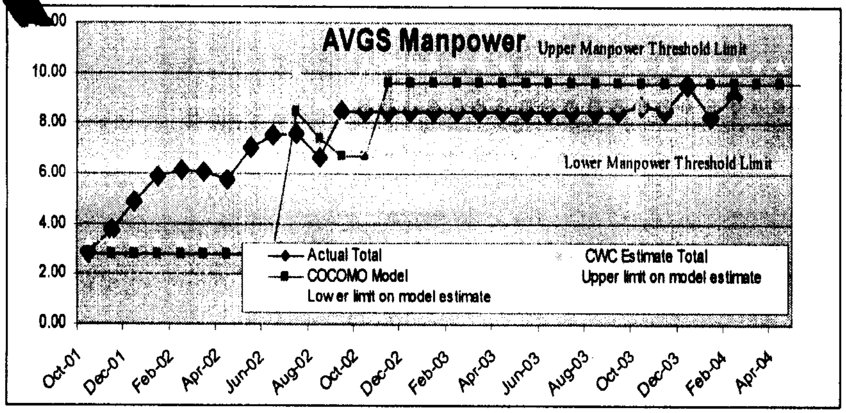
The Software Team collects metric data:
Manual Collection using EXCEL
Automated Collection using PVCS Tracker

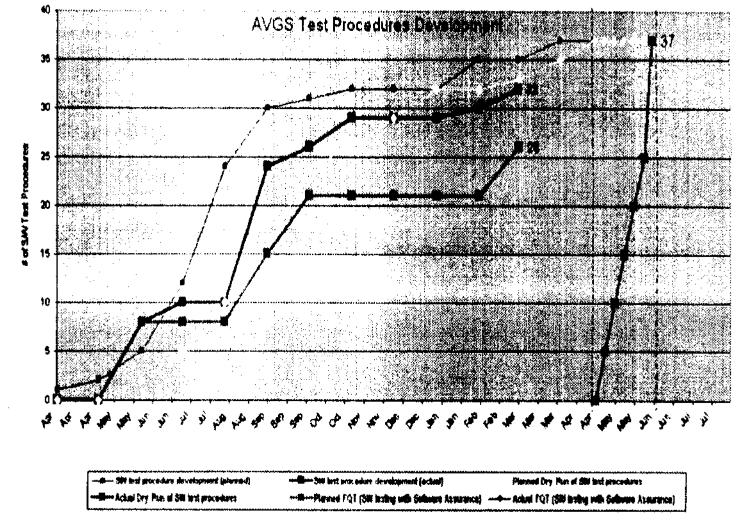
- Computer Resources
- Program Size - SLOC
- Stability – Requirements
- Program Size – Coding
- Program Size – Testing
- Software Change Request

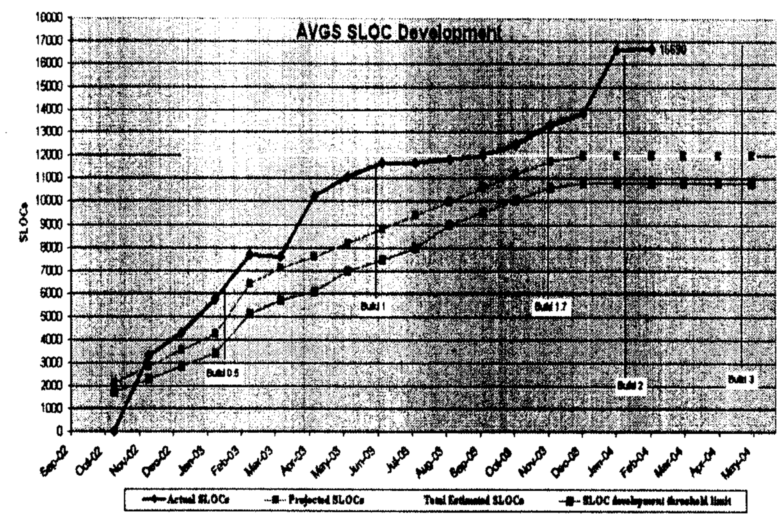
Software Project Lead reports the project metrics to the FSG Metrics Coordinator.
Reports are generated and analyzed

The FSG Metrics Coordinator archives Metrics Data and Documentation in the Process Asset Library (PAL)

Metric Trend Analysis







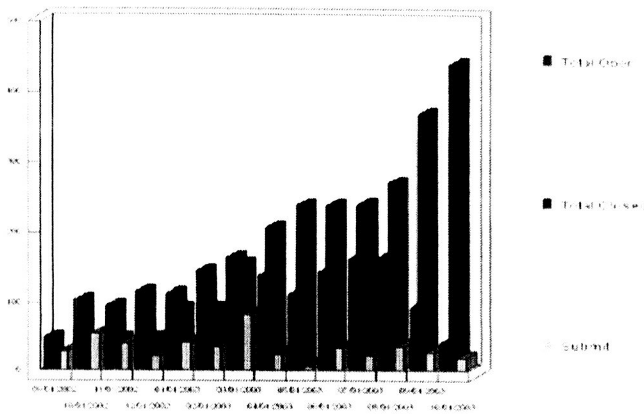
MSFC's Software Trend Charts

Software development manpower trends
Software requirements stability trend
Software Source Lines of Code Development Trend
Software test procedures development and execution trends
Software Change Request Status trends
Software Change Request Identification Methods
Defect Origination by Lifecycle Phase
Number of Defects per CSC
RIDs and Software Problem Reports (SPRs) trends

Defect Reports and Trending

MSRR SCR's bar chart

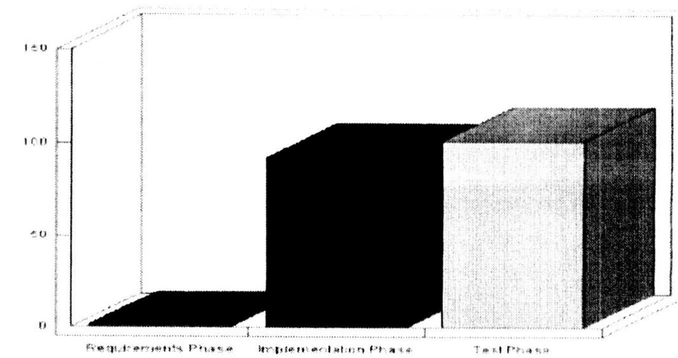
SCR-MSRR Records



09/01/2002 to 10/01/2003

UPA Lifecycle Phase

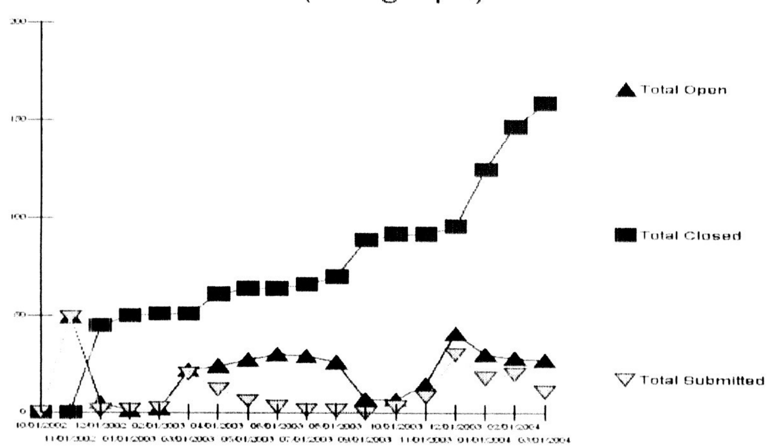
SCR-UPA Records



Lifecycle Phase

AVGS SCR's (line graph)

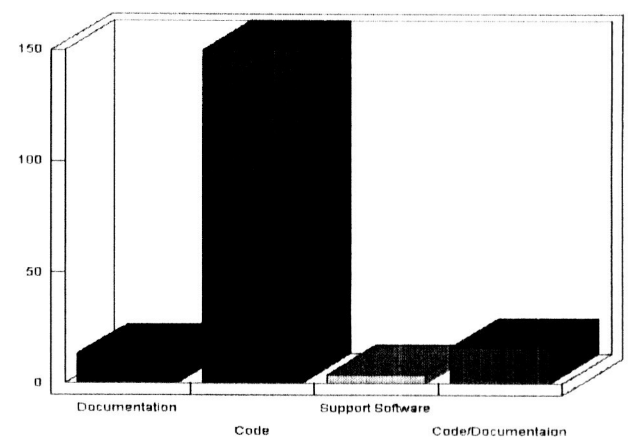
SCR-AVGS Records



10/01/2002 to 03/01/2004

Category of Defects

SCR-AVGS Records



Category

Space Flight Software Best Practices

- Fault case issues shall be addressed and solutions incorporated into the design as early as practical
- The flight software shall be designed to support measurement of computing resources, such as throughput and memory
- The software self-test and built-in test routines shall be removable for flight.
- The information system design shall have engineering emergency data modes and formats (measurements) for diagnostic use.
- Flight software loads/updates and sequence memory loads, particularly for those affecting mission critical capability, shall be verified by a memory readout or checksum readout.
- The telemetry system end-to-end design shall permit ground operators, early in the ground tracking pass, to determine rapidly and unambiguously the state of the spacecraft, particularly to determine if the spacecraft executed a fault protection response.
- Flight software shall be designed to accommodate processor resets during mission-critical events, such as entry/descent/landing.
- Flight software shall be designed to detect and respond to incorrectly formatted commands, data, or loads, and memory faults allocated to the software, such as stuck bits or single event upsets (SEU).
- Software shall be designed to tolerate and continue functioning in situations where inputs are temporarily missing.
- Software shall be demonstrated to be free of deadlocks.
- Software shall be designed to protect against incorrect use of memory: Execution in data areas, unused areas, and other areas not intended for execution, Unintended over writing of code areas.

Space Flight Software Best Practices

- Software shall be designed to ensure that data sets and parameter lists are consistent with respect to time when passed among processes such as software subsystems, rate groups, and others. For example, software shall not be interrupted in a manner that permits it to use both old and new components of a vector.
- Shall be the responsibility of any organization proposing to procure off-the-shelf software to document, prior to procurement, the plan for certifying that such software can be assigned the same level of confidence that would be required of equivalent software obtained through a "development" process.
- The System shall assess the use of dissimilar redundancy in the design of critical functions, as a defense against common cause failure.
- The flight crew should be able to override automatic initiation sequences.
- Software safety shall be an integral part of the overall system safety and software development efforts associated with human rated space flight systems.
- The control of vehicle flight path and attitude, during dynamic phases of flight such as ascent and entry, shall be provided by independently developed and redundant software systems.
- Use of Independent Verification and Validation (IV&V)
- Electronic access to all software products
- Uniform Coding Standards
- The contractor and subcontractors associated with S/W development responsibilities shall be at Software Engineering Institute Software Capability Maturity Model Level II or higher.

Software for Intelligent System Health Management (ISHM)

Briefing For
IEEE Computer Society, UAB
October 20, 2004

Luis Trevino, Ph.D.
Advanced Sensors & Health Management Systems Branch
Spacecraft & Vehicle Systems Department
Engineering Directorate, MSFC

Briefing Agenda

Overview of Health Management Paradigms

ARC – Houston Software Eng. Tech. Workshop April 20-22, 2004

Identified Technologies Relevant To Technical Themes of ISHM

Thoughts

Overview of Health Management Paradigms

- Health Management (HM) technologies determine health of components / systems and subsystems for the purpose of informed-decision making either with humans in the loop or via Intelligent Autonomous Control
- Applicable to all aspects of space exploration – launch vehicles, CEV, upper stages, insertion/ascent stages, planetary habitats, etc:
- Technologies contributing to health management capabilities include:

  1) Advanced software algorithms, models, and software development technologies
  2) Fault Detection Diagnosis (including discrimination between component failures, sensor failures, internal software anomalies, actuator failures and nominal transients) , and recovery (or mitigation)
  3) Prognostics – the estimation of remaining life
  4) Information Fusion
  5) Degradation Management
  6) Smart Data Compression
  7) HM Technology Design Tools – HM needs to be incorporated as an integral part of the design process rather than an add-on. This will require a paradigm shift
  8) Software dependability (health management of software)

Overview of Health Management Paradigms

Autonomy and Intelligence **(Per H&RT SISM Team)**

- Autonomy is a combination of three attributes:
  1. Task complexity
  2. Robustness to unexpected circumstances
  3. Level of human commanding
- Any program or device that can perform complex tasks in changing or incompletely known environments with little human oversight is by this definition autonomous. Thus from a systems engineering point of view, autonomy should be considered for any task that is non-trivial, is performed in an environment that cannot be fully predicted or controlled, and for which human oversight is limited or unavailable.
- This last criterion, the unavailability of human oversight, plus the finite speed of light, are the fundamental source of NASA-unique autonomy requirements – no other agency, and generally no private companies who are not working for NASA, need to perform complex tasks far enough from earth that detailed human oversight becomes impractical
- Intelligence pertains to the ability of devices and systems to be able to perform complex tasks robustly with limited human oversight (life support, power, propulsion, etc.)

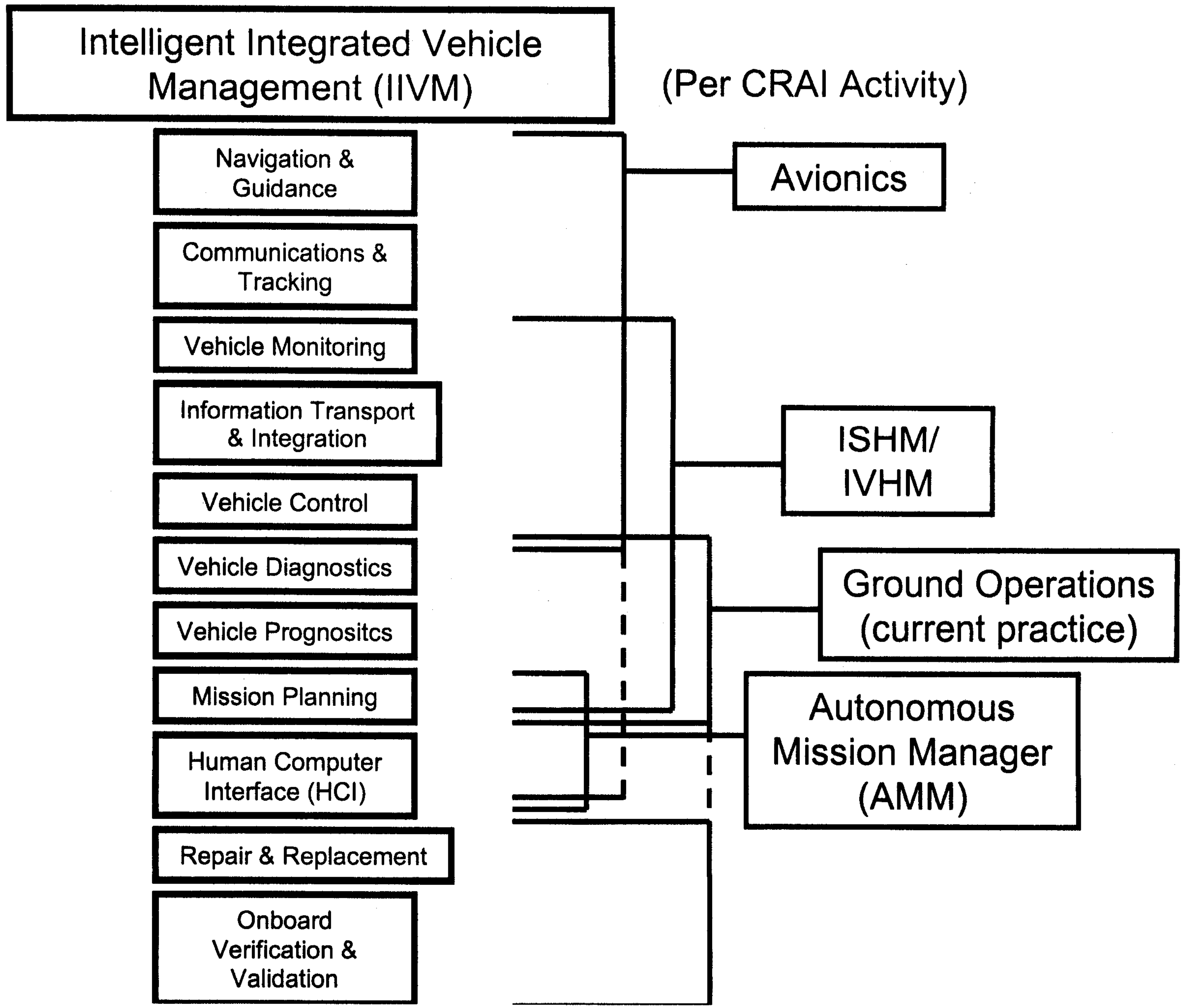
Overview of Health Management Paradigms

1. **Intelligence Enables Safe Vehicle Operation (Per CRAI & MSFC Activities)**
   a. Greatly increase Crew Safety and Mission Success
      1) Vehicle can continue to sustain crew & meet mission objective during communications disruptions
         a) Critical factor in going to Mars
         b) Eliminates crew safety dependence on remote communications capabilities
      2) Automated functions respond to unexpected events in milliseconds, manual onboard functions respond in minutes, ground functions can take minutes to hours.

2. **Intelligence Minimizes Crew and Vehicle Size**
   a. Autonomy allows small crew sizes to safely operate complex vehicle functions
   b. Enables smaller vehicles
      1) Reduced crew size affects living space volume, consumable storage, life support systems

3. **Intelligence Minimizes Ground Operations Staff**
   a. Autonomy increases crew safety and mission success
   b. Autonomy reduces current ground staff which are expensive to operate
   c. Autonomy reduces issues with variance in Martian vs. Earth day cycle
   d. Ground based flight support
      1) Maintain status of mission for NASA and public
      2) Distribute Science Information
      3) Provide engineering support in the event of major vehicle failures

Overview of Health Management Paradigms



Overview of Health Management Paradigms

- Vehicle Level Management Functions
  - GN&C
  - C&T
  - Mission Planning
  - Vehicle Control
- Subsystem Focused Management Functions
  - Monitoring
  - Diagnostics
  - Prognostics
  - Subsystem Control
- Subsystems: Propulsion, Flight Control, Structural, Electrical Power, Thermal Management, Crew Environment, Robotics, & Payload

ARC – Houston Software Engineering Technology Workshop April 20-22, 2004

- Participants from DFR, LaRC, ARC, MSFC, GRC, JPL, JSC, & FAA
- Used IVHM as example to address 2010 & 2024 gaps and needs
  - Today primarily a mixture of subsystems of Health Management
  - Monitoring, diagnostics, prognostics, trending (dealing with degradation)
  - Rigorous semantics: be able to reason about behavior (SE Tools)
  - Verification & Certification
  - Big Issue / Gap: How to <u>build, test, & trust</u> (incremental process)
  - Standard components defined in a way that enable auto verification
  - Levels of abstraction must not inhibit depth of diagnosis
- "Technology will come up with a good idea" (Dan Cooke)

Identified Technologies Relevant To Technical Themes of ISHM

Real Time Intelligent Software Elements List

- Software health management: self-monitoring, self-configuring healing and recovery, self validating (usage of interlocks, fail-safe, self checking mechanism techniques, model-based approaches)
- Model-based software fault recovery and software fault avoidance
- Real Time Onboard data mining and software trend analysis
- Enhanced on-board data storage, processing, and data retrieval (including data compression techniques, data integrity and quality, spacecraft as a web server, IP data routing, secure access)
- Advanced architecture & frameworks for Software

Identified Technologies Relevant To Technical Themes of ISHM

Real Time Intelligent Software Elements List

- Onboard mission and maneuver planning, execution, attitude control, and collision avoidance
- Intelligent machine / human relationships: Natural language high level task uploads, interface to direct science and spacecraft goals & priorities
- System Real-time and health monitoring and automated fault detection, avoidance, isolation, & recovery
- Advanced software techniques to address Single Event Upsets

Identified Technologies Relevant To Technical Themes of ISHM

Real Time Intelligent Software Elements List

- Dynamic on-board reconfiguration of flight software
- System and Software Real-time performance tuning
- Enhanced software voting techniques
- Automated sensor & actuator calibration and integration
- Partitioning between mission and non-mission critical applications

Identified Technologies Relevant To Technical Themes of ISHM

Intelligent Software Engineering Tools

- Software analysis tools including software reverse engineering tools, static software analysis tools, and real-time software analysis and verification tools (e.g., prediction of software defects and of future software system trajectories & validation envelope penetration)
- Software practices for COTS certification and integration
- Utility & certification of auto-generated code tailored to NASA software from design specs
- Generic software simulators / test beds
- Methods for V&V of software systems (Model-based autonomous, intelligent, adaptive flight control, etc.)

Identified Technologies Relevant To Technical Themes of ISHM

Intelligent Software Engineering Tools

- Methods to automate the verification & regression testing of software, its interfaces, and its test procedures
- Device independent interface software
- Software assurance practices for reused / heritage software
- Life cycle robustness, especially for new applications (emerging paradigms and algorithms): need better or combined lifecycle models for reliable software development and test indicators and metrics
- Graphical and readable software representation tools (graphical modeling languages)

Identified Technologies Relevant To Technical Themes of ISHM

Intelligent Software Engineering Tools

- Software risk assessment tools
- Software requirements hazard analysis – fault tree analysis
- Software requirements capture
- Rapid prototyping to explore mission software requirements and design specifications
- Personnel management: process, tools, etc. offset software personnel turnovers
- New software languages & techniques (e.g., objective oriented, real time applications, Java, etc.)
- Libraries of standard components for development & reuse

Identified Applications Relevant To Technical Themes of ISHM

Agency Wide Activities:

- CRAI (Capabilities Requirements Analysis Integration)
  - Concepts: IAHM&C, IIVM, IVM
- NGLT
- ISHM Agency Wide Working Group
- NASA SWG – Software Technology Infusion Strategy Three
  - Collaborations Involving ARC, JSC, JPL, IV&V, MSFC, LaRC
- Collaboration with ARC on automated software analysis & verification tools
- Others (e.g.): JSC Architecture Study, Mars Reference Mission, OASIS

Space Shuttle Main Engine Avionics Technologies

- Improved methods for software testing of mission critical software systems
  - Automated Offline Test Generation Technology
  - Improved methods for software testing of mission critical software systems
  - Generic Simulation Technologies
  - Software Analysis Technologies
  - Expansion of the parameter simulation/patching Technologies
  - Real-time Data Reduction/Analysis Technologies
  - Real-time Data Display/Analysis Technologies
- AHMS Phase IIB

Thoughts

- **Software**
  - Intelligent Software Engineering (ISE) is needed: Tools, Test & Verification Platforms, Efficient Development Processes, automation, results interpretations, risk mgmt, requirements, etc.
  - Ties health management systems all together
- **Intelligent Systems (IS)**
  - Needed for more autonomous operations, crew safety, and mission success
  - Autonomous operations includes reconfigurability, diagnostics, & prognostics
- **Modeling**
  - Improved formal methods & mathematical model development needed for further supporting ISE for IS
- **Universal Theory** (Standard) of ??IVHM?? And corresponding discipline needed

  - HM paradigms must exhibit situational awareness & docile features
- **Verification & Validation** (Certification) Technologies need to progress to accommodate larger state space of possible test scenarios

Questions ???